Horn節制約プログラム上のMAP推定

著者	橋田　浩一
雑誌名	制約に基づく日本語の構造の研究
巻	10
ページ	167-182
発行年	1996-01-31
その他のタイトル	Horn setsu setsuyaku puroguramu jo no MAP suitei
URL	http://doi.org/10.15055/00005485

Horn 節制約プログラム上の MAP 推定

橋田 浩一

概 要

制約 (情報の流れを捨象した記述のレベル) に基づく情報処理システムの設計における主要な研究課題は、全体として妥当な計算が行なわれることを保証する方法である。ここでは、隠れマルコフモデルや確率的文脈自由文法などの一般化として Horn 節プログラムにアナログ的な優先度を与え、これに関する MAP 推定 (最大事後確率推定; maximum a posteriori probability estimation) のための効率的な計算法を提案する。特に、素性構造などの処理の効率を高めるための構造共有の方法に関して論ずる。制約に基づく文法はほぼ Horn 節プログラムによって記述可能であり、この方法によって効率的な処理が期待される。

A major research issue in designing information-processing systems based on constraint (level of description abstracting away from information flow) is how to guarantee global adequacy of computation. We consider Horn clause programs annotated with analog preference, which is a generalization of Hidden Markov Models, Stochastic Context-Free Grammars, etc., and propose an efficient computational method for MAP estimation. In particular, we discuss structure-sharing for the sake of efficient treatment of feature structures, among others. Constraint-based grammars can be encoded in Horn programs, and hence expected to be efficiently processed by the proposed method.

キーワード: 制約 (constraint) 、MAP 推定 (maximum a posteriori probability/preference estimation) 、構造共有 (structure-sharing)

1 はじめに

情報の流れをこと細かく記述する手続き型の設計法に従う限り、複雑な情報の流れを実現しようとするとシステムが複雑化して管理不能になり、設計が破綻する。人間なみに柔軟な情報処理が機械にできていないのは、こういうわけでさまざまな種類の情報の間の複雑な相互作用を実現できないからである。そこで、仕事の手順を具体的にこと細かく設計するのを

やめて、情報の流れをいっさい明示しない設計、つまり**制約** (constraint) を用いることを考える。制約に基づく方法では、設計の段階で破綻することはないにしてもそれによって実現される計算は極めて複雑なものになるから、ここでの主要な研究課題は、いかに情報の流れを制御して全体として妥当な計算が行なわれるようにするか、という問題である。

本稿では、制約に基づく設計において全体として妥当な計算が行なわれることを保証するための一般的な方法について述べる。それは、1階のHorn節プログラムによって表現された制約の意味を解の候補の確率分布としてとらえ、MAP解 (最大事後確率解; maximum a posteriori probability solution または最大事後優先度解; maximum a posteriori preference solution) を求める探索法である。この枠組は、隠れマルコフモデル (HMM) (Xuang, Ariki, & Jack 1990) や確率的文脈自由文法 (SCFG) や確率的木接合文法 (STAG) (Shabes 1992) など、広く用いられている既存の確率的方法の自然な拡張になっている。それら既存の方法では確率的に独立な事象しか扱っていないが、以下で論ずる主要な問題は、説明 (解の候補) の間の構造共有から生ずる確率的依存関係の扱いである。

制約に基づく文法理論はほぼHornプログラムとして表現できる。また、SCFGやSTAGを一般化するような形でHorn節に確率を与えるのは簡単だから、これによって制約に基づく文法に確率を与え、文脈依存性を理論的に捉えることができると考えられる。しかし、一般に制約に基づく文法理論は素性構造 (feature structure) のような複雑な対象に関する制約を含むので、そのプログラムの内容は単なるCFGやTAGではなくなり、SCFGやSTAGなどに関する処理方法によって効率的に処理することはできない。上記の構造共有は、素性構造の処理を含む広い範囲の計算において有効であり、確率的制約に基づく文法理論の効率的な処理方法を与えることが期待される。

以下ではまず2節で基本的な概念を導入し、3節において構造共有を用いた効率的な記号計算法について述べ、次に4節で包摂化を用いてMAP解を求める方法を与えた上で、5節で計算の例を示す。

2 優先度付Horn節プログラム

以下では例として

(a) $\leftarrow p(A) \wedge q(A)$.

(b) $p(X) \leftarrow X=f(a)$.

(c) $p(Y) \leftarrow Y=f(b)$.

(d) $q(Z) \leftarrow Z=f(U) \wedge r(U)$.

(e) $q(W) \leftarrow W=g(V) \wedge r(V)$.

(f) $r(a)$.

(g) r(b).

のような 1 階の Horn 節プログラムを考える。大文字で始まる名前は変項を表わし、それ以外の名前は述語または関数を表わす。p(X) のように述語の後に 0 個以上の項を並べたものを**要素式** (atomic formula) と言い、Z=f(U) のように変項と関数適用を等号で結んだんものを**束縛** (binding) と言う。a のように項のない関数適用は定項である。束縛の左辺の項を**束縛項** (bound term) と言う。2 つの束縛の束縛項が等しければ関数適用も等しい。たとえば、X=f(U) ∧ X=f(V) ならば U=V でなければならない。また、X=f(U) ∧ X=g(V) は常に偽である。要素式と束縛とをまとめて**リテラル** (literal) と呼ぶ。各節は、その中に表われるすべての変項に関して全称量化された命題を表わす。各節をその左辺のリテラルの述語の**定義節** (definition clause) と言う。各述語の意味はそのすべての定義節から作られる必要十分条件によって定義される。たとえば上記のプログラムの下では、r(X) ならば X は a または b でなければならない。

自然言語の文法、特に統語論は Horn 節プログラムによってほぼ記述可能である。たとえば、(1) に示した 2 つの文脈自由規則はそれぞれ、(2) に示した 2 つの Horn 節によって符号化される。

(1) $P \rightarrow Q\,R$
$P \rightarrow a$

(2) p(X,Z) ← q(X,Y) ∧ r(Y,Z).
p(U,V) ← U=a(V).

p(X,Z) は、語列上の位置 X から Z までが P という非終端記号によって支配される句だという意味である。したがって、前の方の節は、「X から Y までが Q で Y から Z までが R ならば X から Z までは P である」という意味になる。また、U=a(V) は、位置 U から次の位置 V の間に語 a が生起しているという意味である。したがって、後の方の節は、「語 a は単独で P である」という意味になる。素性構造等に関する制約のプログラミングも可能であるが、詳細は省略する。

議論の便宜のため、以下ではしばしばプログラムを図で表わす。たとえば前記のプログラムは図 1のように図示される。各閉曲線に囲まれた部分が節を表わす。節の中にある結線は、X などの項を表わす。節の外を通る結線を**単一化結線** (unification link) と呼ぶ。単一化結線は項同士またはリテラル同士の単一化可能性を表わす。2 つのリテラルが単一化結線で結ばれているときは、それらの対応する項が単一化結線で結ばれており、逆も真であるとする。2 つの要素式 (およびそれらの項) の間の単一化結線を**融合結線** (resolution link) 、2 つの束縛 (およびそれらの項) の間の単一化結線を**因子化結線** (factoring link) と呼ぶ。因子化結線は初期状態においては存在せず、後述のような計算の進行に伴って作られる。

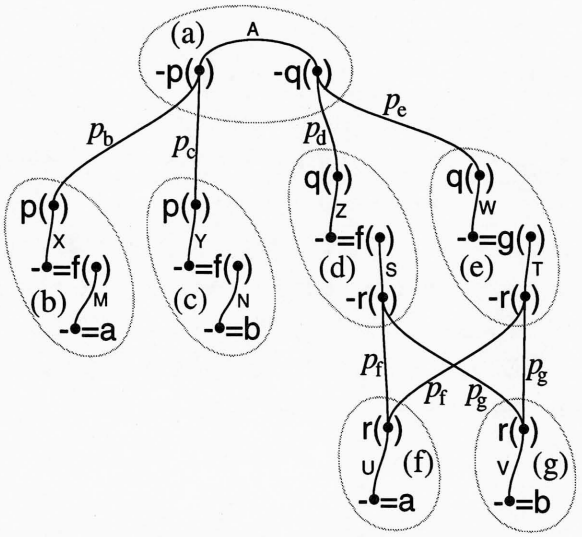

図1: プログラムの図による表示

要素式および束縛の連言を**仮説** (hypothesis) と呼ぶ。各節の前件 (右辺) は1つの仮説である。後件 (左辺) を持たない節を**先頭節** (top clause) と言う。先頭節の前件が、証明すべき最上位の仮説である。上記のプログラムの場合、(a) の否定 $\exists A\,\{p(A) \wedge q(A)\}$ が最上位の仮説である。(先頭節が複数個あるときは、いずれか1つが表わす仮説を証明すればよいとする。) 前述のように各述語の意味はその定義節に基づいて必要十分条件によって与えられているので、p(•) の解集合は {f(a), f(b)} であり、q(•) の解集合は {f(a), f(b), g(a), g(b)} であるから、このプログラム全体の解集合は {f(a), f(b)} となる。

仮説の**説明** (explanation) とは、融合 (resolution) によってその仮説を束縛の連言に変換する仕方のことである。図2に上記のプログラムの先頭節の説明8個のうちの4個を示す。説明を記号的に表示する場合は、そこで用いる節を下降的かつ左から右へという融合の適用順序[1]で並べたものによって表わす。たとえば、上記の仮説の説明のうち、f(a) という解に対応するものは、⟨b, d, f⟩ である。一方、⟨b, d, g⟩ という説明は矛盾を含む。単に「説明」と言った場合は、(いずれかの) 先頭節の前件の説明を意味する。プログラムは説明の集合を表わすと考えられる。

図1に示した通り、各節 Φ の左辺のリテラルにつながった融合結線に正定数 p_Φ が割り当てられており、これを Φ の**基本確率** (basic probability) と言う。1つの説明の確率をそれが含む節 (の具現例) の基本確率の積とする。たとえば、説明 ⟨b, d, f⟩ の確率は $p_b p_d p_f$ であ

[1]この議論は各節の前件のリテラルの間に左右の順序を仮定しているが、この順序は便宜上のものであり、実際の計算には無関係である。また、以下で述べる計算は融合ではない。

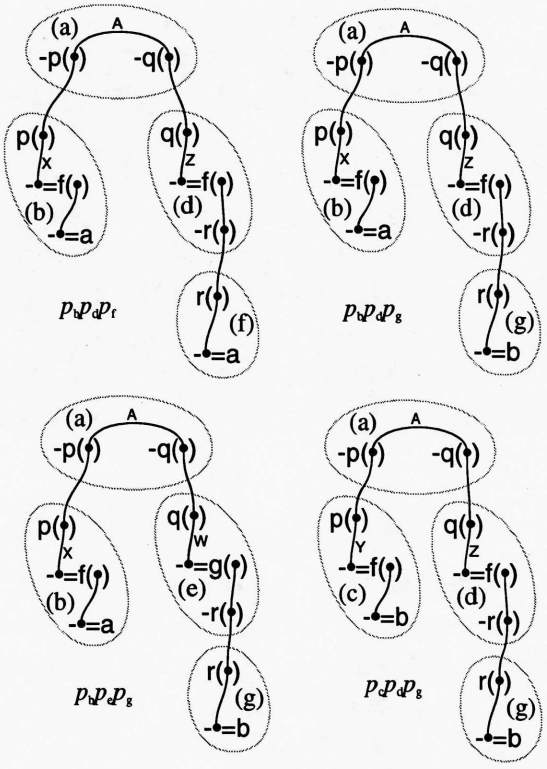
(a)
A
-p(•)
-q(•)
p(•)
x
-•=f(•)
(b)
-•=a
q(•)
z
-•=f(•)
(d)
-r(•)
r(•)
(f)
-•=a
$p_b p_d p_f$
(a)
A
-p(•)
-q(•)
p(•)
x
-•=f(•)
(b)
-•=a
q(•)
z
-•=f(•)
(d)
-r(•)
r(•)
(g)
-•=b
$p_b p_d p_g$
(a)
A
-p(•)
-q(•)
p(•)
x
-•=f(•)
(b)
-•=a
q(•)
w
-•=g(•)
(e)
-r(•)
r(•)
(g)
-•=b
$p_b p_e p_g$
(a)
A
-p(•)
-q(•)
p(•)
Y
-•=f(•)
(c)
-•=b
q(•)
z
-•=f(•)
(d)
-r(•)
r(•)
(g)
-•=b
$p_c p_d p_g$

図 2: 説明の例

る。図2には各説明にその確率を付記してある。各述語ごとにその定義節の基本確率の和を1以下に正規化しておけば、各仮説の解の確率の和が有限な値に収束するから、これに正規化を施すことにより、プログラムの意味を解の事後確率分布として解釈することができる。(「事後」というのは、「入力を先頭節として与えた後」という意味である。) このような確率的解釈が必要なのは基本確率に関するパラメタ学習を行なう場合などであり、以下で述べる計算法においては確率的解釈は必須ではない。

以下では、このようなプログラムに基づくMAP推定 (MAP estimation) の方法、すなわち、与えられた条件の下でMAP解を求める計算法について論ずる。この計算においては、Prologのようにトップダウンとか左から右へとかいう処理の方向性があらかじめ固定されておらず、文脈に依存してさまざまな方向への情報の流れが生ずる。この意味において、プログラムは制約 (constraint) として振舞う。

MAP推定を行なうには、目下のところ確率が最大の説明について、それが矛盾を含むかどうかを調べ、矛盾した説明にしか含まれないプログラムの部分を消去してゆけばよい。最初に見付かった無矛盾な説明が1つのMAP解である。その際、計算の効率を上げるため、異なる説明の間でかなり徹底的に構造共有を行なうことにより、なるべく多くの説明を一挙に調べられるような方法 (Hasida 1994) を用いる。本論文で述べる主要な技術的成果は、そのような構造共有の方式の下で確率最大の説明を効率的に同定する方法である。特に、単なるCFGやTAGなどの場合に限らず、素性構造に関する制約を含む複雑な制約をも効率的に処理する方法について述べる。

3 記号計算

プログラム中の2つの項の間に**依存関係** (dependency) があるとは、それらの項 (の具現例) がある説明において単一化されることだとしよう。そうした単一化は、図1のようなプログラムのネットワーク中の項の経路によって媒介される。そのような経路をその依存関係の**依存経路** (dependency path) と言う。たとえば図1においては、項XとWの間の依存関係が依存経路X·A·Wによって媒介されている。

記号計算の目的は、主として矛盾を含む説明を除去することと考える。説明における矛盾とは、2つの矛盾する束縛項 (たとえば•=f(•) と•=g(•) の左辺) がその説明の中で依存経路によって結ばれていることである。このような依存経路を矛盾した依存経路と呼ぼう。図1のX·A·Wは矛盾した依存経路である。

すると、矛盾した依存経路を除去する計算を行なえばよいことになる。それには、そのような依存経路の通り道にある節や結線を消去する必要があるが、それによって解が失われないようにするためには、矛盾を含む説明のみに含まれる部分だけを消去しなければならない。たとえば、図1の節 (e) はいかなる解にも含まれないので消去してもよいが、(b) は解 ⟨b, d, f⟩ に含まれるので消去してはならない。

解と矛盾した説明が重なり合っている可能性があるとき、矛盾した説明だけを除去するため、**包摂化** (subsumption) という演算を用いて一種のプログラム変換を行なう。包摂化とは、単一化を一般化したものである。プログラムの各部がその基礎具現例 (ground instances; たとえば f(h(a,g(b))) のような、変項を含まない項) の集合を表わすと考えて、項ξと項ηについて$\xi \supseteq \eta$であることをξがηを**包摂** (subsume) すると言い、さらにξが束縛項であるとき、ξをηの**原点** (origin) と言う。項 (の集合) δによる包摂化は、δを原点とする項を生成してゆく演算である。δを包摂化の原点と言う。

この包摂化は、δから出発して、δを原点とする項からなる依存経路を作ってゆく。その依存経路上の項τが生成されたとき、τがδと矛盾しない束縛項ならば、τとδに対する 2 つの束縛を因子化結線で結ぶ。一方、τがδと矛盾する束縛項ならばτを消去する (または初めから生成しない)。こうして矛盾した依存経路をすべて除去することができ、かつ解が失われることはないように、以下で包摂化の仕方を定める。ただし、紙面の都合により正確な詳細と形式的な正当性の証明は省略する。

項 (の集合) δによる項σから項τへの包摂化の様子を図3に示す。δはσの原点であってτの

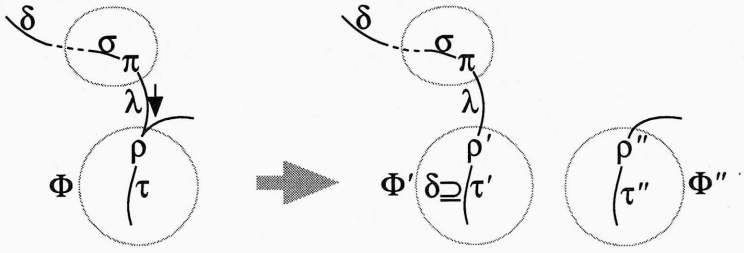

図 3: δによるσからτへの包摂化

原点ではない。また、σを引数とするリテラルπと、τを引数とするリテラルρとは、単一化結線λによって結ばれているとする。(そのようなπとρとλの組が複数個あってもよい。)

節 Φ'はτを含む節 Φ と同形であり、τと対応する Φ'中の項τ'の原点の集合はτの原点の集合にδを加えたもので、Φ と Φ'の他の対応する項の原点の集合は等しいとする。Φ'のリテラルでρに対応するものをρ'とすると、この包摂化により、λはρ'とπとを結ぶ単一化結線に置き換わる。その結果、$\tau' \supseteq \tau \cap \sigma$となり、また、包摂化の後の$\tau$を$\tau''$とすると、$\tau'' = \tau - \sigma$である。$\Phi'$がプログラム中に存在していない場合には包摂化によって Φ'を新たに作ることになる。そのような場合の包摂化を**展開** (unfolding) と言い、そうでない場合の包摂化を**畳込み** (folding) と言う。[2] 展開においては$\tau' = \tau \cap \sigma$であり、畳込みにおいてはτ'に$\tau \cap \sigma$が加

[2]本稿で述べるプログラム変換はたとえば Tamaki and Sato (1983) のそれとは変換の単位が異なるので、「展開」や「畳込み」の意味もそれに応じて異なる。

わる。また展開においては、Φの中のρ以外のリテラルに繋がった各単一化結線は複製されて、Φ'の中の対応するリテラルに繋がる。これにより包摂化は説明の集合を保存する。

以上の計算法によって、たとえば、文脈自由文法などに基づく統語解析を多項式オーダーの計算量で行なうことができる。これは、文法規則に相当する節の具現化が、入力語列に含まれる項からの包摂化のみによって行なわれ、したがって、入力語例中の語数をn、1つの節に含まれる項の個数の最大値をMとすれば、包摂化によってできる節の個数が$O(n^M)$となるためである。さらに、節を部分的に複写する手法を用いることにより、文脈自由文法の場合、空間計算量は$O(n^2)$に、時間計算量は$O(n^3)$になる (Hasida 1994)。これは、よく用いられる効率的な統語解析アルゴリズムの計算量に等しい。

しかし、素性構造などに関する一般的な制約を処理する際には、これだけでは効率が悪い。詳細は省略するが、以上の方法の下では、たとえば、2つの正規言語の間の共通部分を求める計算は無限ループに陥るし、素性構造に関する計算においても構造のコピーを多く作り過ぎる。そこで、コピーは包摂化によって行なわれ、包摂化は異なる原点ごとに行なわれるから、効率を高めるには原点の個数を小さく抑える必要がある。束縛項δが包摂化によって複数個の項 (それらをδの要素と言う) に分割されることがあるが、

(*) δのどの要素についても、それと融合結線で結ばれている項の集合は同じである。[3]

という条件が成立すれば、それらの項の各々を包摂化の原点とする必要はなく、それらの和集合をまとめて1つの原点δとして扱うことにより、包摂化の原点の個数の増大を抑制し、δを原点とする項を含むリテラルや節を構造共有して計算を効率化できる。4節で述べる確率の計算においてこの条件 (*) を用いる。$i \geq 2$に対して束縛の第i引数同士の間の因子化結線を通る包摂化によって直ちに (*) が成立しなくなることはない。

4 確率

ある節を含む確率最大の説明の確率をその節の**最大確率** (maximum probability) と呼ぶ。MAP解を求めるには、最大確率が最大の節を処理するような包摂化を行なっていけばよい。そのような包摂化が実行不能になったとき、最大確率が最大の節から構成される説明が束縛の間の矛盾を含まなればそれはMAP解である。以下では、包摂化によって維持される構造共有の下で各節の最大確率を効率よく計算する方法を示す。計算の進行に伴って節や単一化結線が消去されたとき、この計算法によって各節の最大確率を効率よく再計算することができる。

各リテラルαは**最大内部確率** (maximum internal probability) I(α) と**最大外部確率** (maximum external probability) E(α) を持つと考える。1つの説明は節を節点とする木の形をしており、αを含む各説明はαの所でαを含む部分と含まない部分との2つに分割できる。こ

[3] この条件は少し弱めることができるかも知れない。

れらのうちαを含む部分の確率 (節の基本確率の積) の最大値がαの最大内部確率であり、αを含まない部分の確率の最大値がαの最大外部確率である。節の最大確率はその中のリテラルの最大外部確率の積である。図4に示すように、リテラルの最大内部確率とは同じ節の中

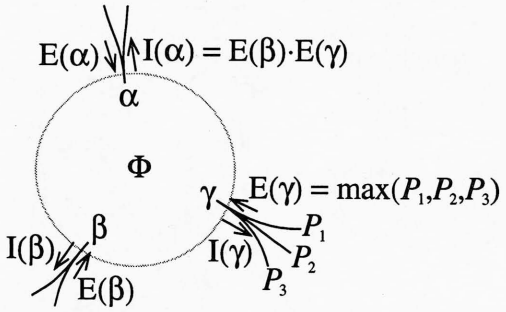

図 4: 確率の間の関係

の他のリテラルの最大外部確率の積であり、これは節の最大確率をそのリテラルの最大外部確率で割ったものに等しい。リテラルの最大外部確率は、そのリテラルと単一化結線で結ばれた他のリテラルからの**接続確率** (connection probability)[4] の最大値である。ただし、束縛αの第1引数が、他の束縛項の原点 (の要素) でなく、かつ他の束縛項を原点とする項でもなければ、E(α)=1 である。リテラルαからβへの接続確率 C(α, β) は、αとβが融合結線で結ばれていれば (つまり、一方が節の左辺で他方が節の右辺にあれば) 、両者を結ぶ融合結線の基本確率とαの最大内部確率との積である。αとβが因子化結線で結ばれていれば (つまり、両方とも節の右辺にあれば) 、これらのうちいずれの左辺の項も条件 (*) を満たす項δの要素である場合はやや複雑なので次節で例を用いて説明するが、そうでない場合は C(α, β)=1 であり、素性構造を持たない CFG や TAG の処理はこの簡単な場合に含まれる。

5 例

図1のプログラムにおける各リテラルの最大内部確率と最大外部確率を図5に示す。太線は最大の確率 $p_c p_d p_f$ を持つ説明を表わすが、この説明は矛盾を含んでいる。図に示したように、$p_b < p_c$ とすれば、

$$\begin{aligned} E(p_a) &= \max(p_b I(p_b), p_c I(p_c)) \\ &= \max(p_b, p_c) = p_c \end{aligned}$$

[4] 一般には、束縛されていない項からその原点への接続確率も考える必要があるが、5節の例題では不要なので、詳細は割愛する。さらに、以下ではどの項の原点もたかだか1つと考える。これによって一般性は失われない。

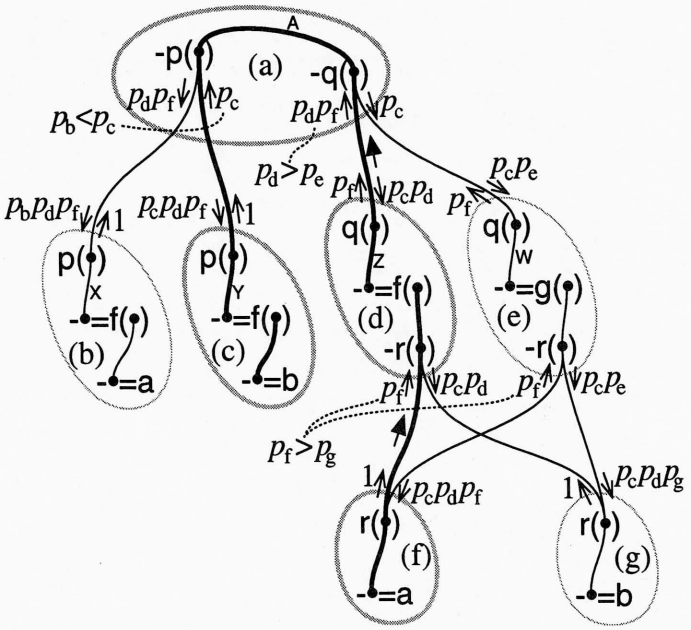

図 5: Z と a による包摂化の起動

であり、$E(q_a)$、$E(r_d)$、$E(r_e)$ に関しても同様である。ここで、リテラルはその述語名 (束縛の場合には関数名) に節の添字を付けて表わし、リテラルの符号は省略する。この説明に含まれる節に対する包摂化としては、(c) から (a)、(d) から (a)、および (f) から (d) に向かうものが可能であるが、ここでは後の 2 つが実行されると仮定する。先端が黒い 3 角形の矢印が包摂化を表わす。

節 (e) はいかなる解にも含まれないが、このことを検出するには、W による包摂化を進めて行くか、単に W から項の融合結線を辿って行けばよい。ここでは、Z による包摂化に伴って自動的に W による包摂化も行なわれるので、それによって W と X および W と Y の間の矛盾が検出され、こうして、W を原点とする A の具現例およびそれを含む (a) の具現例が消去されると考えてもよい。いずれにせよ、今後 (e) はどの説明にも含まれないので無視する。

図5に示した 2 つの包摂化を行ない、さらに Z による包摂化を (b) と (c) まで進めると、図6のようになる。(f) から (d) への包摂化によって (d) が (d1) と (d2) に分割され、Z が Z1 と Z2 に分割されているが、Z1 と融合結線によって結ばれた項も Z2 と融合結線によって結ばれた項も A1 だけだから、依然として Z=Z1∪Z2 をまとめて原点として扱うことができ、Z1 による包摂化と Z2 による包摂化を別個に行なう必要はない。また、(f) から (d) への包摂化に伴い、(g) から (d) への包摂化も同時に行なわれたと考える。

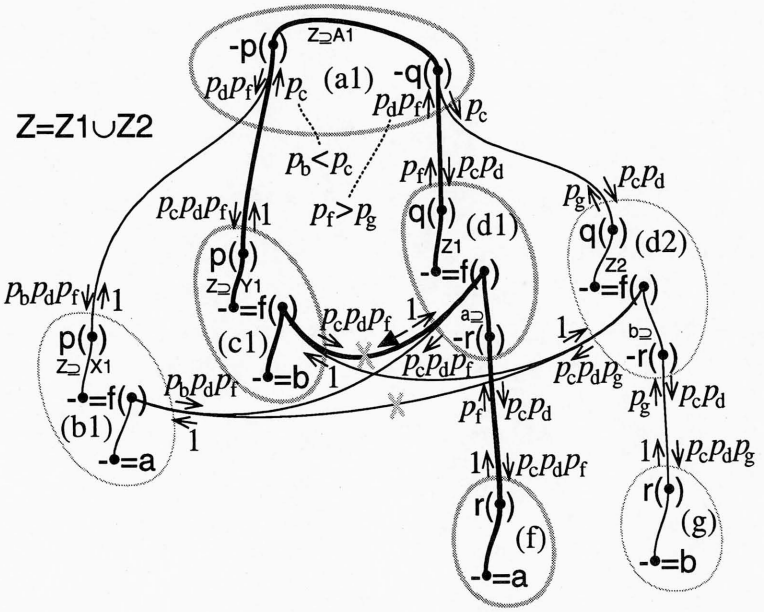

図 6: Z と a による包摂化の結果

本稿では簡単のため、ある項を原点とする包摂化を起動したときは、その瞬間に存在していた単一化結線を通ってその包摂化を最大限まで一挙に進めるものと仮定する。[5] この場合は Z による包摂化が (c) だけでなく (b) にも及ぶと考える。これにより、f_{d1}、f_{d2}と f_{b1}、f_{c1} の間の因子化結線ができる。

図 6のように、δの各要素とδを原点とする各束縛項の間の因子化結線がすべて存在する場合には、δの各要素とδを原点とする各項とを結ぶ各依存経路を含む説明がすべて存在する。したがって、束縛の外部確率をすべて 1 として、束縛の最大外部確率が確率の計算に影響を与えないようにすれば、確率は正しく計算される。

図 6の f_{c1}と f_{d1}の間の因子化結線と f_{b1}と f_{d2}の間の因子化結線は a と b による束縛の間の矛盾を含むので消去される。[6] すると、δの各要素とδを原点とする各項とを結ぶ各依存経路を含む説明がすべて存在するわけではなくなる。つまり、説明 ⟨b1, d2, g⟩ と ⟨c1, d1, f⟩ が消えて ⟨b1, d1, f⟩ と ⟨c1, d2, g⟩ が残る。

この場合は、下記のように、束縛の最大外部確率が 1 でないことがある。まず、δの要素

[5] この仮定は緩和することができるが、その場合は注 4で触れたように接続確率の計算法を一般化しなければならない。

[6] そのような因子化結線の消去に伴って他の因子化結線を消去すべきことがあるが、それはδが 2 引数以上の要素式の引数であった場合だけであり、目下の例題には関係ないので、本稿では考えない。

と融合結線によって結ばれたリテラルのうちで最大確率が最大のものをζとし、

$$\begin{aligned} P &= \mathrm{I}(\zeta) \\ Q &= \mathrm{E}(\zeta) \\ M &= PQ \end{aligned}$$

と置く。現在の例ではζは $\mathrm{q_{a1}}$ である。こうすると、δの要素とδを原点とする束縛項が因子化結線で結ばれている場合、これら 2 つの項に対する 2 つの束縛の最大内部確率の積は、それらを含む確率最大の説明の確率 (つまりこの因子化結線の最大確率) と M との積となる。これは、その確率最大の説明がζを含むからである。つまり現在の例では、

$$\begin{aligned} \mathrm{I(f_{b1})I(f_{d1})} &= M\mathrm{p}\langle \mathrm{b1, d1, f} \rangle \\ \mathrm{I(f_{c1})I(f_{d2})} &= M\mathrm{p}\langle \mathrm{c1, d2, g} \rangle \end{aligned}$$

となるが、これは

$$\begin{aligned} \mathrm{I(f_{b1})} &= Q\mathrm{p}\langle \mathrm{b1} \rangle \\ \mathrm{I(f_{d1})} &= P\mathrm{p}\langle \mathrm{d1, f} \rangle \\ \mathrm{I(f_{c1})} &= Q\mathrm{p}\langle \mathrm{c1} \rangle \\ \mathrm{I(f_{d2})} &= P\mathrm{p}\langle \mathrm{d2, g} \rangle \end{aligned}$$

だからである。したがって、このような 2 つの束縛αとβ (現在の例では $\mathrm{f_{b1}}$ と $\mathrm{f_{d1}}$ または $\mathrm{f_{c1}}$ と $\mathrm{f_{d2}}$) の内部確率の積を M で割れば、これらの間の因子化結線の接続確率が得られる。ゆえに、こうした各束縛の内部確率と外部確率 (その束縛への接続確率の最大値) の積がその束縛の最大確率となるようにするには、各束縛の内部確率を M で割ったものをその束縛から他の束縛への接続確率とすればよい。つまり現在の例では、

$$\begin{aligned} \mathrm{C(f_{b1},f_{d1})} &= \mathrm{I(f_{b1})}/M \\ \mathrm{C(f_{d1},f_{b1})} &= \mathrm{I(f_{d1})}/M \\ \mathrm{C(f_{c1},f_{d2})} &= \mathrm{I(f_{c1})}/M \\ \mathrm{C(f_{d2},f_{c1})} &= \mathrm{I(f_{d2})}/M \end{aligned}$$

とする。こうすれば、プログラムのあらゆる部分においてその最大確率が定まるが、最大内部確率および最大外部確率は必ずしも定まらない。これは、確率の計算が巡回しているためである。この様子を図7に示す。ここで、a、b、c、d はプログラムの各部分の確率である。$ac > bd$ とすれば $M = PQ = ac$ となるが、P と Q の値は決まならない。そこで、内部確

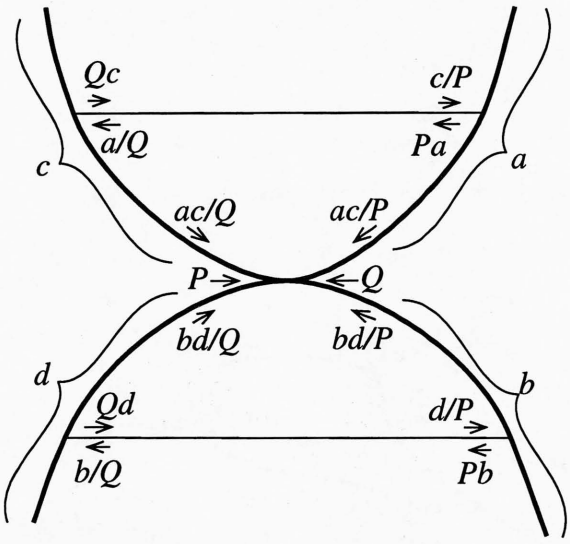

図 7: 確率の計算の巡回

率の積が最大である束縛の組に含まれる束縛の内部確率を Mとすることができる。すると、それらの束縛からの接続確率は 1 となり、最大確率が最大の説明の上では確率の再計算の必要がない。現在の例では、$p_b p_f > p_c p_g$とすれば、f_{b1}と f_{d1}がそれらの束縛であるから、

$$C(f_{b1},f_{d1}) = 1$$
$$C(f_{d1},f_{b1}) = 1$$

となる。以下の計算は省略するが、このようにしてプログラムの状態は図8のようになる。なお、図6の状態において同様の方法を適用すると、各束縛の外部確率はいずれも 1 となる。

リテラルの説明とは、そのリテラルれだけからなる仮説の説明のことだとしよう。図6では、p_{a1}の説明と q_{a1}の説明との任意の組み合せを含むような全体の説明が存在する、という意味において、p_{a1}と q_{a1}とは独立である。しかし図8では、⟨b1, d1, f⟩ と ⟨c1, d2, g⟩ という 2 つの説明しかないので、p_{a1}と q_{a1}の説明の間には依存関係がある。直接間接に呼び出す関係にない 2 つのリテラルの間にこうした依存関係がある場合でも、われわれの方法では、これら 2 つのリテラルの間の構造が多くの説明の間で共有されうる。共有される構造はこの例では (a1)[7] という 1 つの節であるが、一般にはさらに大きな構造が共有されることがある。HMM や SCFG はこうした依存関係が生じない場合を扱っているが、Horn 節プログラムに議論を一般化した場合、特に素性構造を効率的に処理するためには、このような依存関係

[7](a1) は説明には明示的に含まれないが、実質的には共有されていると考えられる。

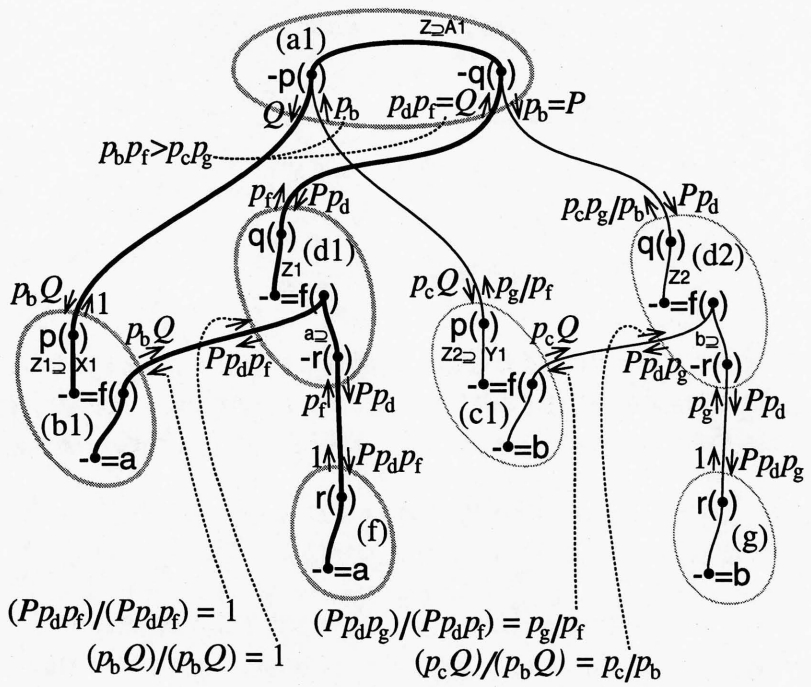

図 8: f_{b1} と f_{d2} および f_{c1} と f_{d1} の間の因子化結線を消去した結果

の下で構造共有を行なう必要がある。

6 おわりに

HMM や SCFG の一般化として確率付 Horn 節プログラムを考え、これに関して効率的に MAP 推定を行なう方法を論じた。この方法は、たとえば文脈自由文法の統語解析を既存のアルゴリズムと同等の効率で行なう一般的な制約処理手続きと、それによって推持される構造共有、さらにそこから生ずる依存関係の下で、プログラムの各部分の最大確率 (その部分を含む確率最大の説明の確率) を計算することにより、一般化された意味での A*探索を行なうものである。本稿では 5 節の例の理解に不要な議論を省略したが、ここで省略した詳細については他の機会に報告したい。たとえば、再帰的なプログラムを再帰的なプログラムに変換することも畳込みによって可能である。

可能な包摂化を尽くせば、すべての説明が解になっており、しかもどの解も失われていないので、プログラムは正しい解の確率分布を表現している。もちろん計算が終了しないこともあるが、説明の確率の総和は計算に伴って単調に減少し、しかも 0 より小さくはならないから、必ず収束する。また、未処理の説明のうちで最大確率が最大のものの処理を優先することにより、正しい解の確率分布への収束が保証される。そのような計算は、基本確率に関するパラメタ学習に用いることができる。パラメタ学習には EM アルゴリズムなどを用いることができる。

本稿で述べた方法は、HMM や SCFG や STAG など、確率が基本確率の積和の形で表わされるいろいろな枠組のうちで最も一般的なものである。これら既存の方法を用いた従来の処理方法は、各作業に特有の具体的な処理手順を用いていたために、たとえば音声認識と言語処理を有機的に統合できなかったが、われわれの方法では作業に固有の処理手順を一切用いず、一様な処理メカニズムから既存のアルゴリズムの挙動が創発するから、多様な情報処理過程を統合し、さまざまな種類の情報の間のより緊密な相互作用を実現できる可能性がある。

また、この方法は、Horn 節プログラムという一般的な記述力を持つ記号体系を用いているので、HMM や SCFG に比べてはるかに広い射程を持つ。たとえば、Bayesian ネットワーク (Pearl 1988) やコスト付アブダクション (Hobbs et al. 1993) などの推論メカニズムをこの枠組の中で実現することができるだろう。また、上述のような方法によるパラメタ学習だけではなく、例に基づく学習や帰納学習による記号的な知識の獲得と組合わせることもできるので、記号的な知識の学習と統計的な情報処理との相互作用を探究する手段となろう。さらに、確率に加えて**効用** (utility) の概念を導入し、期待効用 U を用いて、$\frac{\partial U}{\partial p}$ が大きい基本確率 p を持つ節に含まれる行為を優先的に起動するという行為の理論を考えることができる。効用をどのような形で定義するか、あるいはその場合に情報処理の制御と行為の制御がどのような関係を持つかなどに関して、考察を進めているところである。

参考文献

Hasida, K. (1994). Emergent Parsing and Generation with Generalized Chart. In *Proceedings of the Fifteenth International Conference on Computational Linguistics*, pp. 468–474.

Hobbs, J. R., Stickel, M. E., Appelt, D. E., & Martin, P. (1993). Interpretation as Abduction. *Artificial Intelligence*, **63** (1-2), 69–142.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, Los Altos, CA.

Shabes, Y. (1992). Stochastic Lexicalized Tree-Adjoining Grammars. In Boitet, C. (Ed.), *Proceedings of the Fourteenth International Conference on Computational Linguistics*, pp. 426–432 Nantes.

Tamaki, H. & Sato, T. (1983). Unfold/Fold Transformation of Logic Programs. In *The 2nd International Conference on Logic Programming*, pp. 127–138.

Xuang, X., Ariki, Y., & Jack, M. A. (1990). *Hidden Markov Models for Speech Recognition*. Edinburgh University Press.